Innovative Color Management Methods for RGB Printing

by

Curtis N. Vanderpuije

SB, Mechanical Engineering (2005)
Massachusetts Institute of Technology

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF ENGINEERING IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

AUGUST 2006

Signature of Author.....
Department of Mechanical Engineering
August 22, 2006

Certified by.....
Kamal Youcef-Toumi
Professor of Mechanical Engineering

Accepted by.....
David Hardt
Professor of Mechanical Engineering
SMA program Co- Chairman

Accepted by.....
Lallit Anand
Professor of Mechanical Engineering
Chairman, Department Committee on Graduate Students

Innovative Color Management Methods for RGB printing

by

Curtis N. Vanderpuije

Submitted to the Department of Mechanical Engineering
on August 22, 2006 in partial fulfillment of the
requirements for the Degree of Master of Engineering in
Mechanical Engineering

ABSTRACT

The demand for printing excellent quality images has increased tremendously in parallel to the growth spurts in the digital camera market. Printing good quality images consistently, however, remains a difficult and/or expensive venture despite the numerous advances in color technology and printing. To alleviate these issues, a color compensating software solution was developed to utilize the unique Kikuze calibration chart to improve printer output.

The software solution integrates with the windows printing process at the operating system level through a UNIDRV plug-in. The plug-in retrieves the data within the print stream, passes it on to the color compensation engine which corrects the color data by mapping input and output colors obtained via a B-spline interpolation algorithm. The rendered image is re-introduced into the print stream for final printing. The prototype achieved successful results and can be packaged with commercial printers after a few refinements.

Thesis Supervisor: Kamal Youcef-Toumi

Title: Professor of Mechanical Engineering

ACKNOWLEDGEMENTS

I would like to thank Dr Allan Zhang who worked with us every step of the way, advised us and gave us great ideas. He always made time to meet us and readily explained concepts to aid the development of our ideas. He also encouraged us to explore and try out new ideas.

Thanks to Prof Youcef-Toumi for advising us and keeping us on track to meet the requirements of the project. Thank you for taking the trip all the way to Singapore to meet with us and to ensure that we all had common objectives.

Thanks to Prof Andrew Nee for giving us suggestions on other techniques we could research on as we developed our module. Thank you for taking the time to review our theses and for pointing out areas that needed improvements.

Thank you to Dr Du Xian who was also instrumental in this project, giving us practical and relevant ideas based on his research experience, teaching us the relevant aspects of color science and working with us all the way through to implementation of the color control module.

I also want to thank Ryan Wong for readily sharing his expertise in color and design. He added his professional touch to our user-interface designs and worked with us in developing the test methods for our color control module. Thank you for also making the working environment at Kikuze lively.

Thank you to Mr. Winson Lan for sponsoring this project and giving us the necessary motivation and push to pursue the project all the way through.

Lastly, I would like to thank Dong Wei, who has been a great partner to work with – sharing the joyous moments of discovery, sharing the stressful moments of frustration and for working side by side as a great teammate to the very end.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1 Introduction and Background

The growth and advancement in computer graphics technology and printing devices have made color printing affordable not only to business users but also to home users. The spurt in the development of digital camera technology and the boom in digital camera sales put pressure on printer makers as they try to meet the user demands of photo quality printouts. An investigation by the Gartner Dataquest and International Data Corporation indicates that the printer market is largely driven by the growth in web-based color applications and digital cameras [1]. Digital cameras emphatically increase the use of color printers as everyday users seek to print memorable images. Unfortunately, the printer makers are yet to achieve the same quality output as the traditional printing process for cameras. As a result, many users turn to commercial vendors to print digital images. Camera makers such as Canon have tried to enter the niche market by providing portable custom printers that hook onto the digital cameras to print photo quality images. These portable printers are however not readily affordable to the everyday user.

In addition to the actual quality of the printer output, is the inconsistency across different printers. Printing the same image on two different printers will not always yield the same result in regards to color. Color mismatch is further aggravated by nonlinear characteristics such as gamut mismatch between scanner and printing devices, print head deterioration, printer halftone techniques, cartridge or toner variation, a change in paper stock, temperature, humidity and other environmental fluctuations. These factors affect the interaction of the colorants on the paper and their non-linear relation results in color output variation.

These issues are difficult to deal with and result in highly expensive printers that try to compensate or minimize these variations. Kikuze Solutions seeks to circumvent these expensive solutions and tackle the problem from a software base. The basic approach is to

utilize the patented Kikuze calibration chart to calibrate printers before printing and is further elaborated in the subsequent sections.

1.1 Project Objectives

The project objective is to develop a Color management System to minimize output variances among printers from various venders. The main idea is to intelligently compensate for the color differences with the software solution. Color compensation will be done based on the comparison between the printout and the unique Kikuze calibration chart. The objective can be broken down into the following sub-objectives:

- Figure out the optimal and feasible stage for integration of the software solution with the Windows XP printing architecture (key entry point)
- Use RGB color space instead of CMYK in the existing solution
- Determine work flow of color control module
- Develop and integrate color control module with the existing Kikuze solution
- Test and evaluate work flow and developed solution
- Design user interface
- Develop full working prototype
- Make prototype ready for demos to potential customers
- Extend prototype for automatic color correction

1.2 Research Approach

This research is a continuation of prior work at Kikuze Solutions and will integrate the developed solution with the existing ICCS system. The main difference between the existing solutions is that the current solution is simplified as it is targeted at everyday users.

It should also be noted that this is a team project and the thesis is therefore written in conjunction with "Innovative Color Management for RGB printing" by Dong Wei. Some aspects might require reading sections of both theses for best understanding. Dong Wei's thesis primarily focuses on the integral details in the color compensation engine whereas this thesis focuses on the entire work flow and incorporation in the Windows printing architecture.

Chapter 2 Theoretical Background

This section of the thesis explains the various stages in the windows printing architecture, explores the options available for color management and highlights important aspects of color science and RGB printing.

2.1 Windows Printing Architecture

In order to fully understand the complexity of the project, the printing data process in Windows XP must be understood. The general architecture is shown in Figure 2.1 and will be used to further explain the print process. The Windows DDK documentation and prior Kikuze work are the main sources of documentation for the Windows printing architecture.

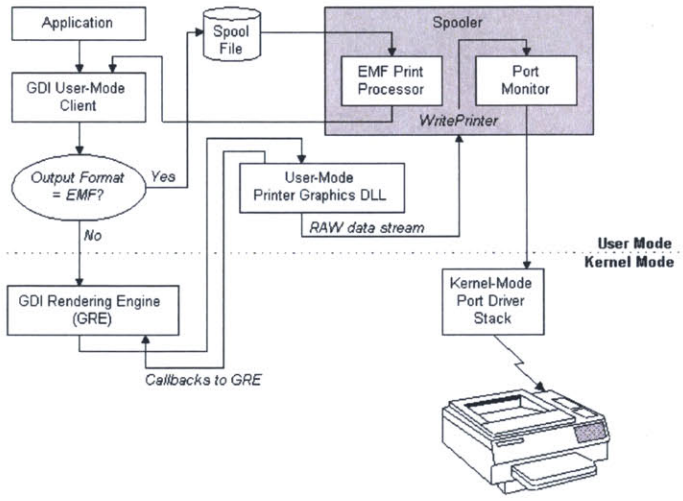

Figure 2.1: Windows Printing Architecture flow [2]

Print jobs created by users run through several processes within the Windows printing system before finally being output by a printer or other output device. The print jobs are sent to the printer via device independent Win32 and Windows GDI (graphic device interface) printing functions. The entire process can be broken down into the following three stages:

2.1.1 The Application stage

At the application stage, a user action to print a file from an application starts by calling the Win32 GDI function *StartDoc*. This command and other commands from the GDI are sent to the GDI graphics engine. The GDI graphics engine can either spool the file immediately or send it off to the GDI rendering engine (GRE) and appropriate printer drivers for further processing. This decision is based on the file format received by the GDI graphics engine. EMF (enhanced meta files) or files in RAW data format can be sent directly to the spooler whereas others will require rendering before spooling.

2.1.2 The Spooler stage

A spooler is optional for most print jobs. Users can choose to use a print spooler or process jobs on their resident computers before sending to printers directly. However, the processes performed at the spooler stage remain the same regardless of where they are carried out. The spooler is one of the main components of the printing process. The spooler performs functions such as locating the correct printer driver, loading the driver, scheduling the print job and adds high level functions to the print job. The spooler creates two files (a spool file and a data file) after the GDI function *StartDoc* is called by the application. The spool file contains the EMF data from the application and the data file contains information about the print job such as the target printer. The spooler renders the print job and then forwards the job to the selected printer.

2.1.3 The Printer stage

The spooler actually sends the print job to a print processor which handles the conversion of the spooled data into printer 'readable' data. The print processor works with the printer driver's graphics DLL to convert the data. At this stage, the data has been rendered in a form that the printer can understand via the GRE and appropriate printer driver. The print processor then forwards the job

to the print monitor which directs the job to the appropriate port driver for the target printer. The printer reads the bitmap commands and prints.

2.2 Printer Driver

A printer driver is a program that provides communication between Windows and a specific printer. Print drivers are responsible for rendering print jobs with custom drawing capabilities that the Windows GDI cannot support. Printer drivers are also responsible for sending rendered image data to the print spooler and also provide an interface for user selections. Print drivers also transfer printer specific information to the Windows GDI to aid the GDI in processing print jobs for the specific printer. The print driver usually consists of two separate DLLs – one for rendering functions and the other for the user interface options (dialog boxes and property sheets). The printer driver also stores files known as characterization files that contain hardware information about the specific printer. In summary, the printer driver works with the GDI graphics engine to convert Microsoft® Win32® GDI calls from applications into device specific drawing functions that the output device can understand.

2.3 Color Management methods in Windows

Color management in the Windows system can be controlled in four distinct ways:

- By the application
- By the Windows GDI
- By the driver
- By the hardware device

The desire to have a universal color management system narrows the control to the second approach – the Windows GDI. Within the Windows GDI, there are two main ways of controlling color management:

- By the use of ICC profiles
- By an operation system level implementation

2.3.1 ICC profiles versus OS level implementation

The ICC approach involves the development of a lookup table and the integration of the solution with the pre-existing ICC and ICM color management systems present in the Windows operating systems. The Windows GDI renders images (device independent bitmaps, pens and brushes) based on existing ICC profiles before sending the images to the printer driver.

The OS level implementation is based on a full integration with the Windows printing architecture. This involves integration with the Windows GDI system and potential print data files such as the spool or PRN files. This approach will incorporate integration with the Windows API and GDI functions as the print data stream is interrupted, the image is processed and re-introduced into the print data stream after color correction. These two approaches are further elaborated in the implementation section.

2.4 Image Processing and the GDI

Image rendering in the Windows operating system is always controlled by the GDI rendering engine. The default color management system in Windows (the ICM) uses the GDI in conjunction with the resident ICC profiles to map input and output color spaces to correct the colors in images. The GDI uses the *IPrintOemUni::ImageProcessing* method with the Unidrv (universal print driver explained in subsequent sections) to modify bitmap images (i.e. perform color management).

2.4.1 Windows GDI

The Windows GDI is responsible for all graphics within the Windows operating system. This includes graphical displays on monitors and image processing for printing. The GDI serves as a link between the Windows system and applications as shown in Figure 2.2.

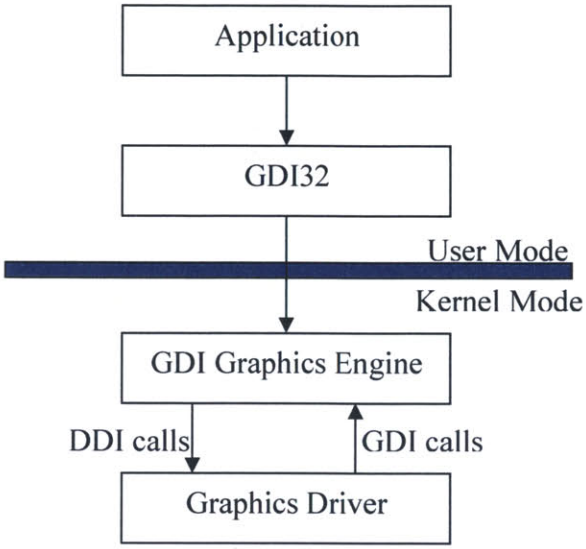

Figure 2.2: Graphical representation of Windows GDI [2]

Evident in the figure, the GDI consists of two main parts:

- The GDI user mode:

This serves as an intermediary link between application calls to Win32® GDI functions (graphical requests) and windows. The GDI receives these requests and forwards them to the GDI kernel mode.

- The GDI kernel mode

This serves as the link between the GDI and other OS devices such as printer drivers, display drivers and other output devices. It communicates via the graphics DDI (device driver interface).

The GDI is designed with several graphical output capabilities and only needs to call on the graphical DDI when it needs functions it cannot handle. Similarly, the driver is designed to meet only the functionalities the GDI lacks therefore simplifies its design and

implementation. The two components are inter-dependent both exporting and importing functions for various tasks.

2.5 Color Compensation Engine

The color compensation engine is the backbone of the color management system. It involves a full understanding of the above-mentioned components and integrates these components to achieve color correction. The color compensation engine is responsible for the actual process of color correction in images after the image is retrieved from the print data stream. It is synonymous to the rendering that takes place within the GRE. The color compensation engine is driven by a compensator algorithm which consists of the following:

- A data structure for creating a lookup table (LUT)
  (RGB and compensator entities of control points)
- A generation of a LUT from a manual input of sampling points or automatic generation of scanned points
- An interpolation that outputs the compensation for pixel indices in the input LUT

The details of the compensation engine are further elaborated in the parallel thesis by Wei [3].

2.6 Universal Driver (UNIDRV)

The universal printer driver (UNIDRV) is a special printer driver. The UNIDRV provides support for non-Postscript printers and has been built by Microsoft as a table-driven printer driver. This simply means that it has the full framework of a printer driver but gives many options or gateways to enable further customization for many printer drivers hence its name - universal. This simply reduces the complexity involved in creating a printer driver as the programming team would only need to add the additional code to generate the desired

output. The UNIDRV provides this option through plug-ins. Like other printer drivers, the UNDRV consists of a user interface component driven by the UNIDRVUI.DLL and a rendering component driven by the UNIDRV.DLL. Figure 2.3 shows the location and interaction of the UNIDRV with the GDI and other DLLs in a simple print job from NOTEPAD.

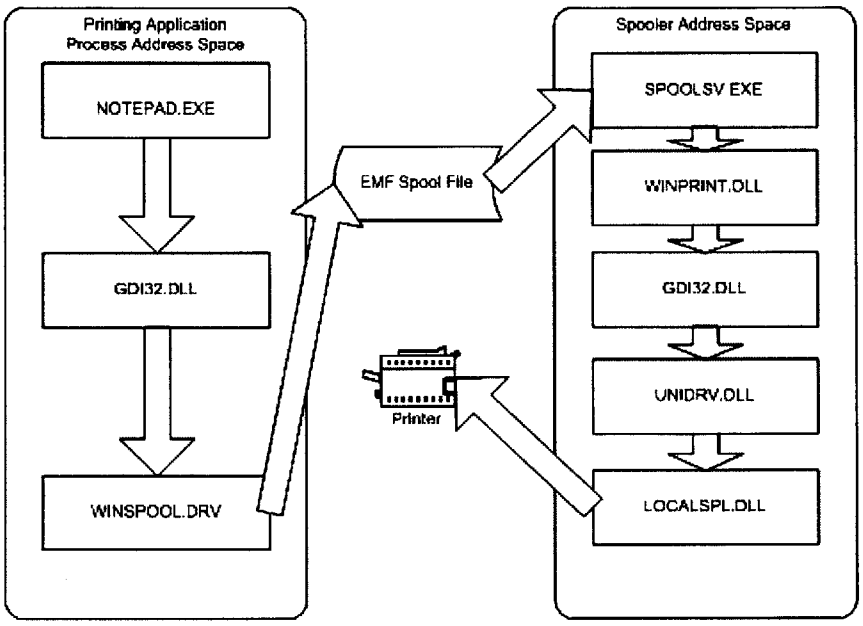

Figure 2.3: UNIDRV in the windows printing system [4]

2.7 Understanding Color and RGB printing

The description of color is subjective since it is based on the perception of the human visual system. Color results from sensation produced on the eye by rays of light reflected or emitted from an object. Through a prism, white light, which consists of radiation at all wavelengths of the visible spectrum, can be separated into red, orange, yellow, green, blue, indigo, and violet. Out of this understanding comes the concept of primary colors.

Primary colors, when combined in different ways, result in a broad range of colors. Many types of primary color sets exist, with the principal sets being additive and subtractive (Figure 2.4: Images of Additive and Subtractive Primaries [5]
). Projected together as beams of colored light, the additive primaries will mix or overlap to produce other colors. The additive primaries are used for a computer monitor or television screen. On the other hand, the subtractive primaries (cyan, magenta, and yellow) are used to create color separations for photography and printing. In subtractive color mixing, pigments such as ink or paint absorb or subtract all the colors of the spectrum except the color that the pigment reflects to the eye.

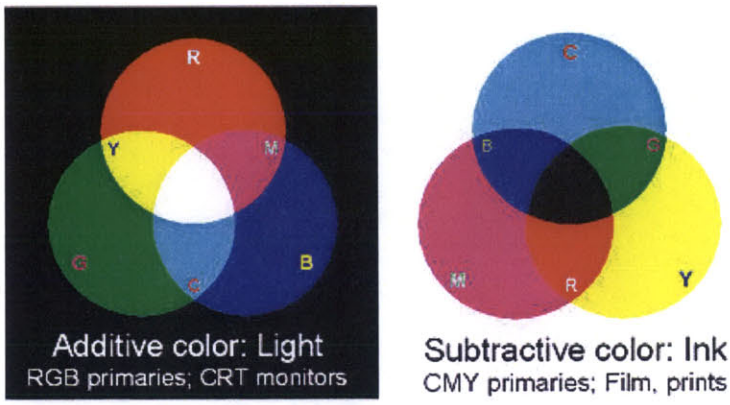

Figure 2.4: Images of Additive and Subtractive Primaries [5]

Color models are methods of organizing the set of possible human color perceptions in a systematic way. Colors or their properties are expressed numerically in these models. A color space is a 3-D geometric representation of the colors that can be produced using a certain color model. Color models can be divided into two main categories, perceptually-based and display-based (device-dependent).

Device-dependent models (or display-based models), such as RGB, are used to create millions of colors on a computer monitor or television screen by combining different values of red, green, and blue. RGB is an example of a device-dependent color model because each device emits a slightly different shade and intensity of red, green and blue light. In the RGB

system each color component of a pixel is measured by a number from 0 to 255, for a total of 256. The RGB model is also used in color scanners and color photography. However, this model does not work well in the printing process.

The CMY (cyan, magenta, yellow) model is a subtractive color model that complements the additive model. CMY model are used in printing because the process is based on light reflecting from colors, such as those contained in a printed image on paper. However, the combination of CMY in full values does not produce a pure black color. Therefore, black (K) must be added to the separations, resulting in the CMYK color model. The CMYK model forms the basis for printing.

Appendix A shows the new chart designed to perform printer characterization for consequent study. The new chart consists of both CMYK and RGB bars, ranging color densities from highlight, midtone to shadow in each row. One unique feature over conventional charts is the key line (line 5) in which black is registered as a combination of CMY rather than printed directly by black ink as in line 4. Comparing the printout with a standard chart, the calibration system can identify the accuracy of the CMYK printing system on RGB image reproduction.

2.8 Chapter Summary

The various stages of printing in Windows have been explained in this section of the thesis. The potential options for color management and existing system implemented by Microsoft have been highlighted. Image processing, the Windows GDI and the UNIDRV have also been illustrated to give a solid background of the underlying basis of the solution module.

Chapter 3 Design and Implementation

This section of the thesis describes the research process in the development of this module. The section covers the various methods that were researched, tried and tested as potential solutions to the task at hand. The section explains the merits of the various approaches and justifies the selection of the final approach. The user-interface designs are presented and the work flow of the final solution is demonstrated.

3.1 Determining the path in Windows

The entry point for the Kikuze Color Control module was one of the main challenges of the project. A brief schematic and the potential entry points are shown in Figure 3.1.

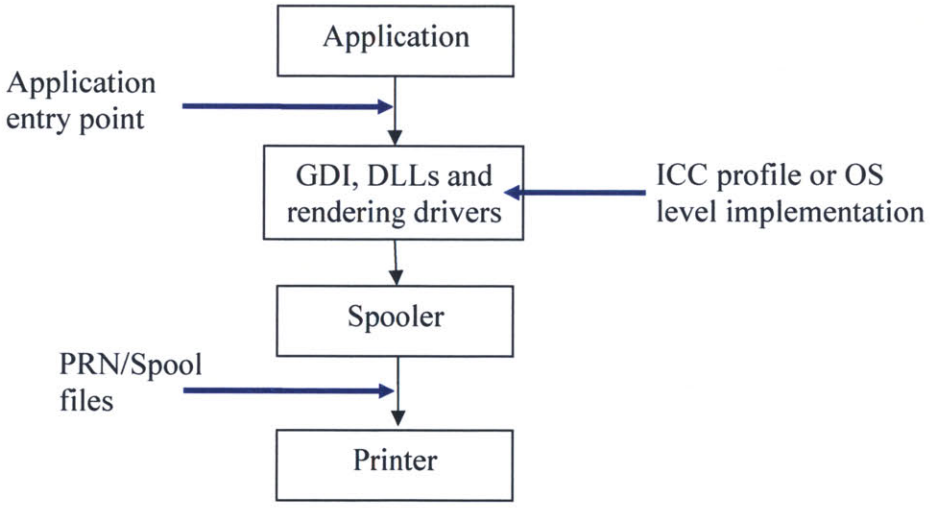

Figure 3.1: Potential Entry Points

3.1.1 The Application Entry point

The application entry was immediately eliminated. This is due to the fact that, the color control module should be application independent. There are too many applications that run in Windows and too many potential outputs into the print stream from these myriad

applications. It would therefore be infeasible to develop a universal color control module that interfaces with many applications.

3.1.2 Spool/PRN entry point

The PRN and spool files were investigated thoroughly as a potential entry point. This was due to the emphasis given to their potential as an entry point for color optimization by prior work at Kikuze. The PRN and spool files have fully processed data that is delivered by the spooler with the main difference being the fact that the PRN has extra data that the spool file does not contain. The format of the image data in both types of files remains the same. In both files, the image data is stored in hexadecimal format. These two file types were treated as a single entry point as the differences between them would not warrant significant differences in their implementation as an entry. The PRN files would, however, be a lot more difficult to interpret due to their dependence on the output device.

To study the nature of these files, the bitmap images (created using MSPaint) and the text shown in Figure 3.2 was printed.

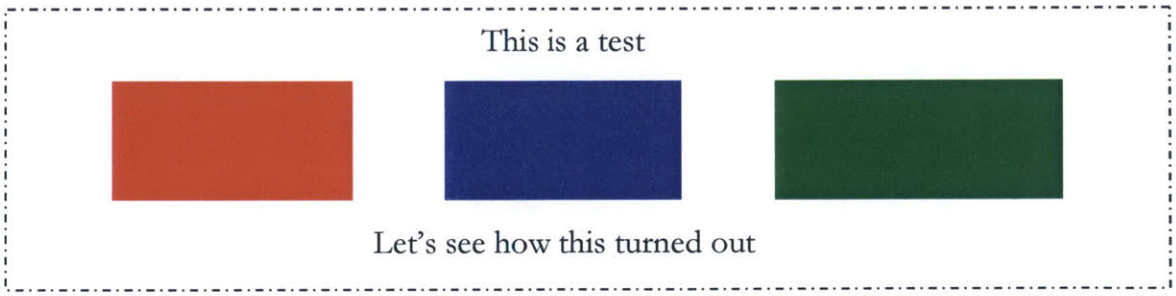

Figure 3.2: Test file for study of Spool file output format

Retrieval of the data files in this approach was relatively easy. It simply requires turning on the option to save the print data to a file. The spooled data could then be saved from any application. At this point, the spooled data could simply be retrieved from the saved location (which is a standard location in the Windows printing system) for color compensation. The file could then be sent directly to the printer for printing. A study of the spool files printed

showed that the spool file was organized in three main sections – the starting stream, data stream and closing stream. Retrieval of the data would have required scanning the starting stream and recognizing the start of the data stream. A study of multiple spool files showed that the starting stream did not always contain the same information and did not have the same length. It varied based on the file being printed, printer information and user options. A sample of the spooled data of Figure 3.3 is shown in.

```
0630d0  00 00 FF 00 00 FF 00 00  FF 00 00 FF 00 00 FF 00  ................
0630e0  00 FF 00 00 FF 00 00 FF  00 00 FF 00 00 FF 00 00  ................
0630f0  FF 00 00 FF 00 00 FF 00  00 FF 00 00 FF 00 00 FF  ................
063100  00 00 FF 00 00 FF 00 00  FF 00 00 FF 00 00 FF 00  ................
063110  00 FF 00 00 FF 00 00 FF  00 00 FF 00 00 FF 00 00  ................
063120  FF 00 00 FF 00 00 FF 00  00 FF 00 00 FF 00 00 FF  ................
063130  00 00 FF 00 00 FF 00 00  FF 00 00 FF 00 00 FF 00  ................
063140  00 FF 00 00 FF 00 00 FF  00 00 FF 00 00 FF 00 00  ................
063150  FF 00 00 FF 00 00 FF 00  00 FF 00 00 FF 00 00 FF  ................
063160  00 00 FF 00 00 FF 00 00  FF 00 00 FF 00 00 FF 00  ................
063170  00 FF 00 00 FF 00 00 FF  00 00 FF 00 00 FF 00 00  ................
063180  FF 00 00 FF 00 00 FF 00  00 FF 00 00 FF 00 00 FF  ................
063190  00 00 FF 00 00 FF 00 00  FF 00 00 FF 00 00 FF 00  ................
0631a0  00 FF 00 00 FF 00 00 FF  00 00 FF 00 00 FF 00 00  ................
0631b0  FF 00 00 FF 00 00 FF 00  00 FF 00 00 FF 00 00 FF  ................
0631c0  00 00 FF 00 00 FF 00 00  FF 00 00 FF 00 00 FF 00  ................
0631d0  00 FF 00 00 FF 00 00 FF  00 00 FF 00 00 FF 00 00  ................
0631e0  FF 00 00 FF 00 00 FF 00  00 FF 00 00 FF 00 00 FF  ................
0631f0  00 00 FF 00 00 FF 00 00  FF 00 00 FF 00 00 FF 00  ................
063200  00 FF 00 00 FF 00 00 FF  00 00 FF 00 00 FF 00 00  ................
063210  FF 00 00 FF 00 00 FF 00  00 FF 00 00 FF 00 00 FF  ................
063220  00 00 FF 00 00 FF 00 00  FF 00 00 FF 00 00 FF 00  ................
063230  00 FF 00 00 FF 00 00 00  0E 00 00 00 14 00 00 00  ................
063240  00 00 00 00 10 00 00 00  14 00 00 00 0D 00 00 00  ................
063250  08 00 00 00 10 32 06 00  00 00 00 00              .....2......
```

Figure 3.3: Sample of spool file for data in Figure 3.2

This example is relatively simple as it shows 00s and FFs indicating a standard red color pixel with RGB values (255 0 0) represented as (00 00 FF = BGR as it reads the data backwards). The spooled representation of colors, however, varies depending on the application the image is printed from. For instance, Adobe Photoshop has a completely different format which is relatively difficult to convert to the standard format [6].

The main drawback of this approach was the fact that we would need to firstly decode the files, compensate for color and re-encode the files to create new spool files. In addition to the application dependence on these files, there are also slight differences in spool files due to printer differences. The combination of these factors led us to investigate other entry point options.

3.1.3 GDI entry point

The GDI interacts with printer drivers through a few DLLs to perform color optimization. The GDI is a great potential entry point. Within the GDI are a few potential methods that can be used to attain color correction.

ICC Profiles

The entry point through ICC profiles involves the current color correction mechanisms employed by Microsoft. The main difference between the existing ICC profiles is the creation of a custom Kikuze ICC profile. This simply acts as another ICC profile within the Windows printing system. This approach would have been relatively easy to implement and could have been developed within a reasonable time frame. The main drawback to using the ICC profile approach was the fact that it was dependent on the current color correction mechanism, with little control by our team to attain custom color correction. It was also susceptible to the inefficiencies within the current Windows printing system. The Windows color management performs conversions due to the use of different color spaces. These conversions can lead to color data loss and we were unsure about the effect on our ICC profile solution after these color conversions are carried out. Indeed, other competitors such as Best Color and GMG have developed an approach based on ICC profiles but these have not attained the most optimum result hence our search for a better approach.

OS level implementation

The OS level implementation approach was the most promising. Considerable time was spent researching and testing sample codes on the MSDN web sites for such development. The *Image Class* was found and it was a potential point for color correction through the Windows GDI. The class had functions that allowed us to create *Color Palettes* which we intended to use as a replacement for color look-up tables (CLUTs) used by ICC profiles. This approach involved switching to C# as the sample code given was written in C#. We managed to transform specific image files via matrix calculation and methods provided by

the *ColorMatrix Class*. A 4 x 4 matrix was used for transformations such as scaling and rotation and a 5 x 5 matrix was used for transformations such as translation. The elements within the matrix represent the intensity and opacity of the colors and range from 0 to 1. The 0 to 1 range corresponds to the 0 to 255 range of RGB colors. A sample matrix is shown below

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0.1 & 0.1 & 0.1 & 0 & 1 \end{bmatrix}$$

In the sample matrix shown, the first three diagonal components represent RGB values, the fourth represents the opacity/transparency of the image and the last diagonal component is left as 1 in the 5 x 5 matrix. The first three components in the bottom row represent translations of color. The matrix shown triples the intensity of the blue color and adds 0.1 to each of the red, green and blue components. Sample code was found from an open source site and tested and the results are shown in Figure 3.4.

Figure 3.4: Color transformation [7]

In this case, the intensity of the red component was changed to 0.9. The original image appears on the left and the corrected image appears on the right. As can be observed, the

entire picture is more reddish. The main limitation in this approach is the fact that the entire image is transformed by the modification. This is not the desired effect in our color control module. The objective is to give color control that enables the user to change the color across the entire 0 to 255 range (i.e. Highlight, Midtone and Shadow).

At the OS level, we further investigated the various DLLs involved in the print process. The *Depends* tool provided by Visual C++ enabled the study of various components within several DLLs. The main limitation was that the use of the functions remained unclear despite finding the output of the functions. Our search was to find out the main DLL that each print process utilizes. Our search ended when we discovered the capabilities of the UNIDRV, its associated DLLs and the rendering plug-in options it provides.

3.2 Color Control via UNIDRV plug-ins

The UNIDRV has two key components for the development of the color control module - the user interface plug-in and the rendering plug-in. These plug-ins integrate seamlessly with the system level Windows printing stream as they are stand alone and designed by the team with total flexibility. The details within the UNIDRV are shown in Figure 3.5 with the relevant components highlighted.

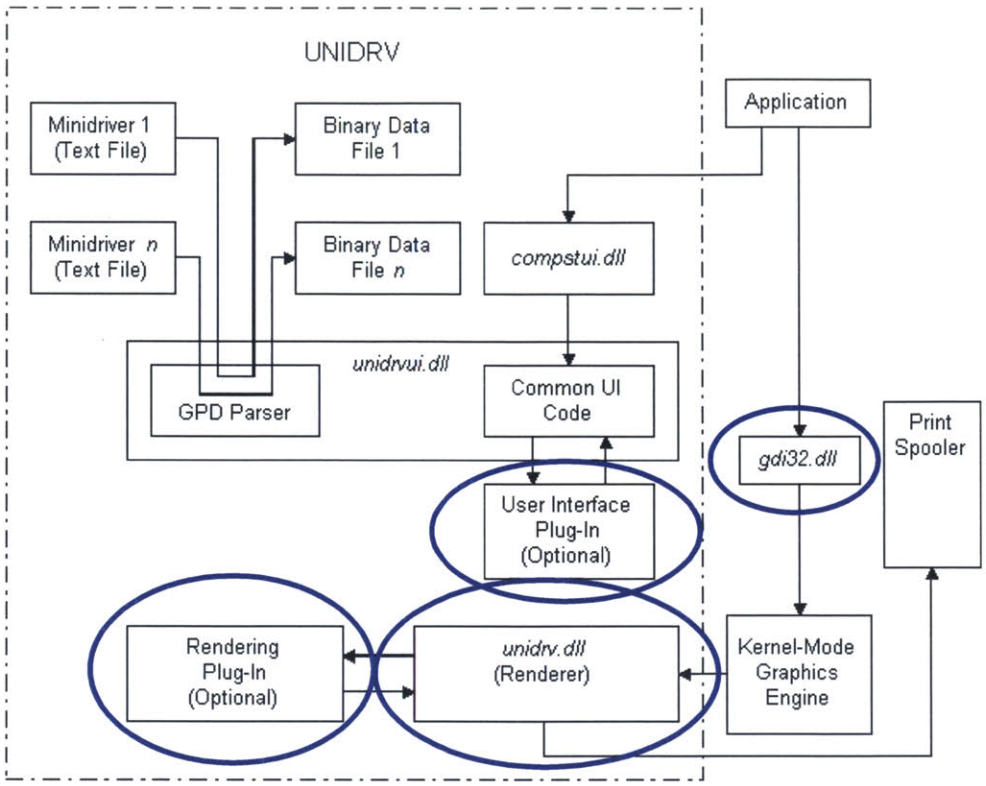

Figure 3.5: Relevant UNIDRV components [2]

The plug-ins are implemented as COM (component object model) in-process servers (DLLs) as the figure shows. The DLLs exports specific entry points such as the *DLLGetClassObject* function which returns a pointer to a COM class factory for the creation of the user-interface and rendering plug-in components. The rendering plug-in has a myriad of functions but the most relevant to the color control module is the color management component.

3.2.1 Rendering plug-in

There are two major functions provided by the UNIDRV that are important for the rendering plug-in: *IPrintOemUni::ImageProcessing* and the *IPrintOemUni::FilterGraphics*. The *IPrintOemUni::ImageProcessing* function performs various color formatting and half toning processes on the bitmap images that are retrieved from the print stream through the UNIDRV. The *IPrintOemUni::FilterGraphics* method is used to modify scan line data before

being sent to the printer. It is used to perform post-processing of image data before spooling.

The framework and work flow of the color control module is therefore established by the use of these functions. Through the *IPrintOemUni::ImageProcessing* method, bitmap images can be retrieved from the print stream (THE ENTRY POINT), processed by the Color Compensation Engine (CCE) and re-introduced to the print stream (EXIT POINT). The UNIDRV rendering plug-in therefore serves as the key entry and exit points for the Kikuze color management module.

By implementing the *IPrintOemUni::ImageProcessing* method, the UNIDRV initiates the Kikuze color management module, each time a fired print job includes a bitmap image. The UNIDRV delivers the address of the bitmap image as an input argument to the to *IPrintOemUni::ImageProcessing* method. The bitmap image is now available for manipulation by the developed CCE. At this point, the CCE should already have the transformation from the color control curves. The CCE can then read the pixel values of the bitmap image and modify the values based on the color transformation matrix. The final output from the CCE is the rendered bitmap image. The rendered bitmap image is then sent to the print spooler by calling the *IPrintOemDriverUni::DrvWriteSpoolBuf* method. The flow through the rendering plug-in is shown in Figure 3.6.

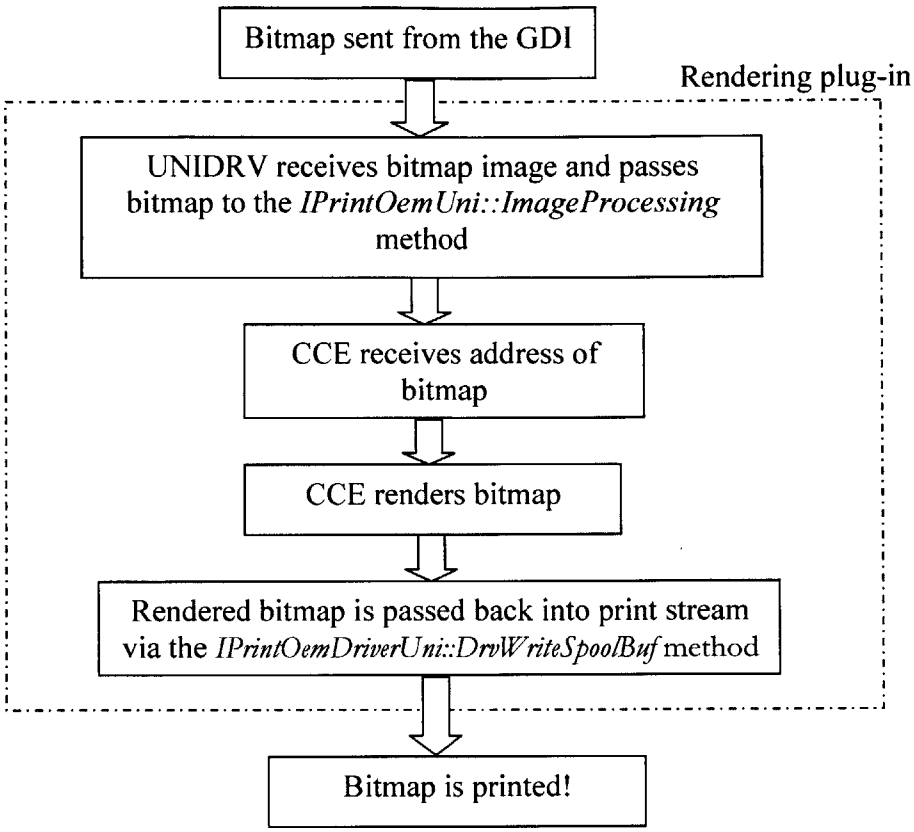

Figure 3.6: Rendering plug-in workflow

3.2.2 User-interface plug-in

The user-interface plug-in is the part of the color management module that interfaces the processes happening within the module with user selections. This plug-in enables customization of the UNIDRV interface, controlled by the UNIDRVUI.DLL. The Kikuze color management module utilizes the user-interface plug-in to add on a device property sheet to the property sheets of a target printer. The device property sheet is modified to contain the elements that enable the user to modify the color control curves to obtain the desired color correction. These elements will be shown and explained in further details in subsequent sections. However, a basic user-interface is shown in Figure 3.7.

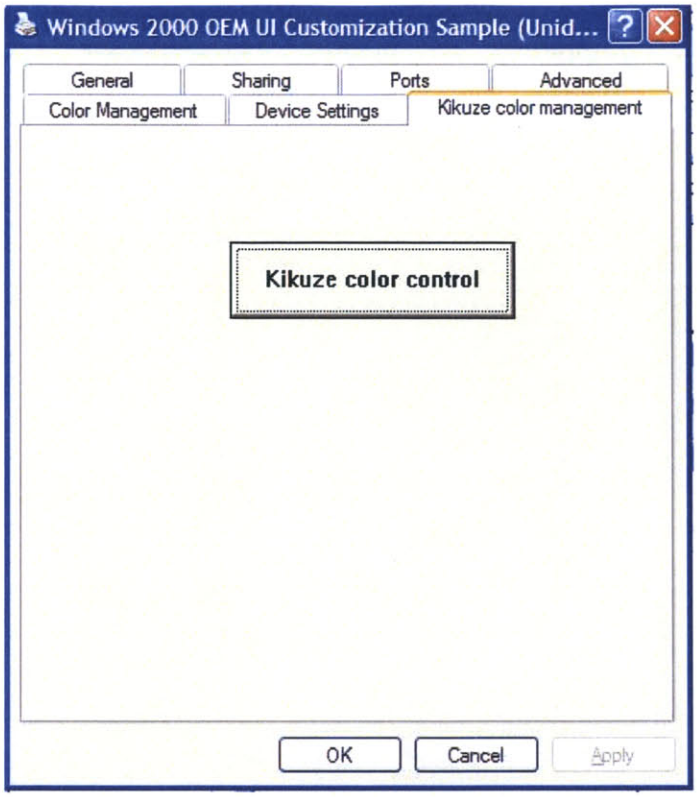

Figure 3.7: User-interface insertion through device property sheets

The active tab represents the sample printer property sheet that will contain the Kikuze color management module. In this simple sample, clicking the button, for instance, will bring up the color control module. The color control module can also be directly placed within this property sheet.

There are several options for the location of the color control module user-interface. The solution shown in Figure 3.7 can be rather complicated and tedious to implement. This is based on the simple fact that the property sheets of all available printers will have to be modified to include the color control module. This will require obtaining the printer *handles* of all the printers and the use of the *IPrintOemUI::DevicePropertySheets* methods to modify the printer property sheets of all the printers. This standard approach might be feasible but is further complicated by the fact that the addition of new printers on the network will not automatically include the Kikuze color control module. As a result, the installation of the color control module might have to be rerun to add the module to newly installed printers.

An alternative approach to avoid re-installation each time a new printer is added is to modify the printer property sheets during runtime – an approach that can be rather tedious and is better suited to professional printer driver developers.

An alternative approach that was investigated by the team involves the operation of the Kikuze Color Control module as a stand-alone application in regards to the location of the user-interface. The application is still fully integrated with the printing process, the only difference being the fact that the user-interface is displayed as a stand-alone application. In this approach, the Kikuze color management module can be simply called from the Windows print dialog box as shown in Figure 3.8.

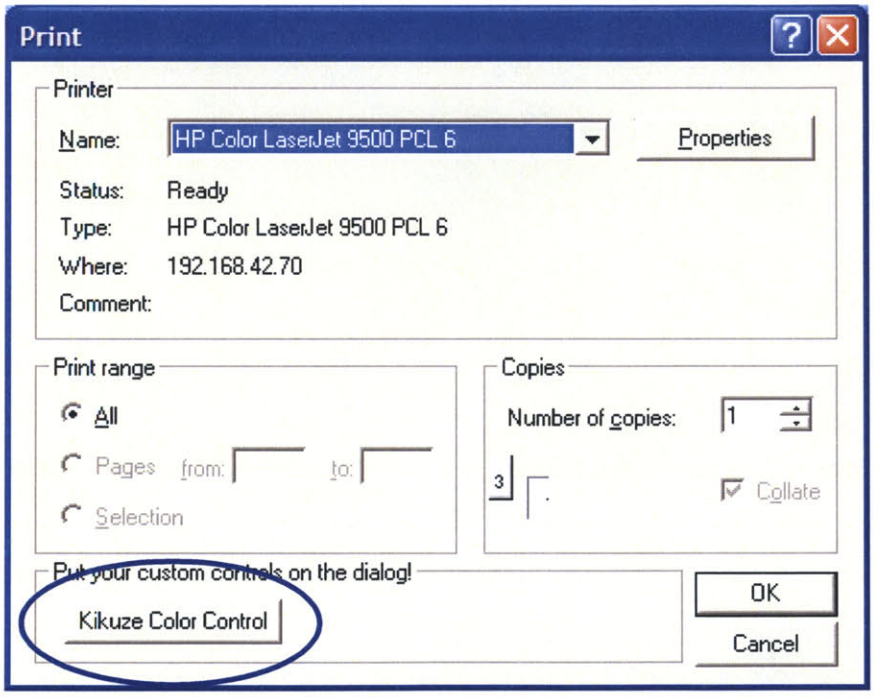

Figure 3.8: Kikuze Color Control module accessed through the Windows print dialog

This approach has one main advantage – the Kikuze color control module can be accessed from any application. The control curves can be reached quickly, modified and the print job sent off to the specified printer.

3.3 User Interface designs and considerations

The Kikuze standard calibration sheet provides the user with the entire range of the RGB color spectrum (i.e. from 0 to 255 in a Highlight, Midtone and Shadow order) for manipulation. As a result, in designing a user-interface, the team considered the thought process a user goes through in modifying pictures to improve the color quality. Firstly, it was imperative to use the same color spectrum on the calibration chart on the user-interface. This was to ensure that the user had a basis of comparison as an everyday user will need to compare the printed output with the standard Kikuze color calibration chart to attain optimum color quality. The first design resulted in the user-interface shown in Figure 3.9.

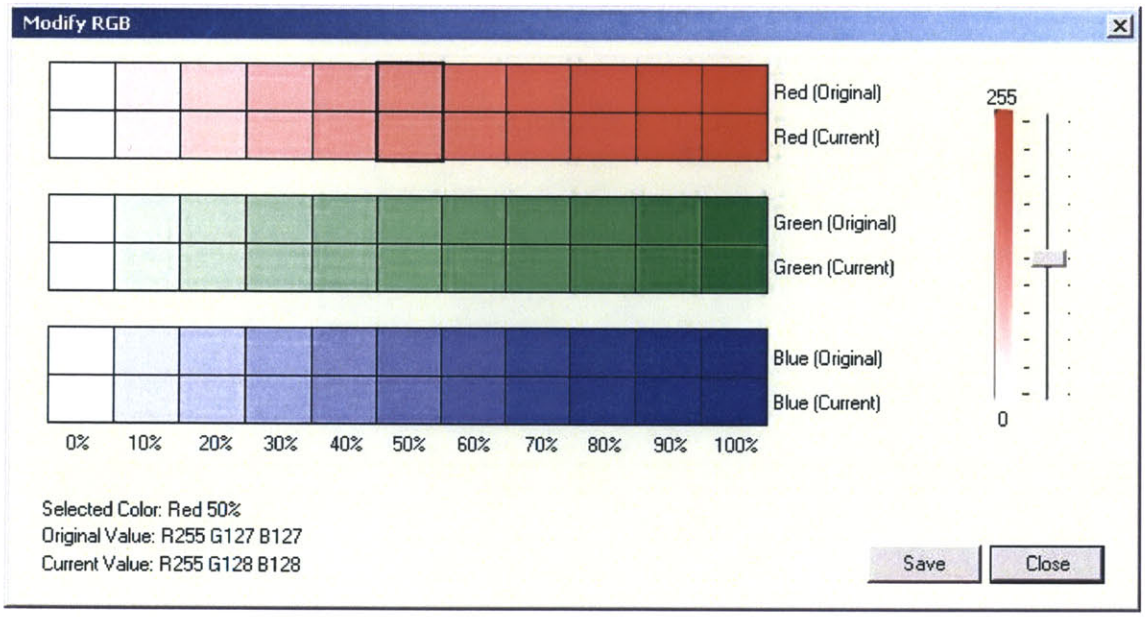

Figure 3.9: First user-interface iteration (Design 1)

In this design, the user can select any of 33 'colors' to modify. These 33 colors correspond to points on the color calibration curve that the color compensation engine (CCE) uses to attain color correction. The user makes the changes by dragging the slider bar on the right side of the interface to adjust the color accordingly. A great feature that has been added to the interface is a standard row which represents the original color and gives a real time

comparison as the user makes changes. The numerical results are shown in the bottom left corner of the interface.

Though this user-interface is relatively simple, it is slightly disconnected from the color correction method and was modified to include the color control curves as shown in Figure 3.10.

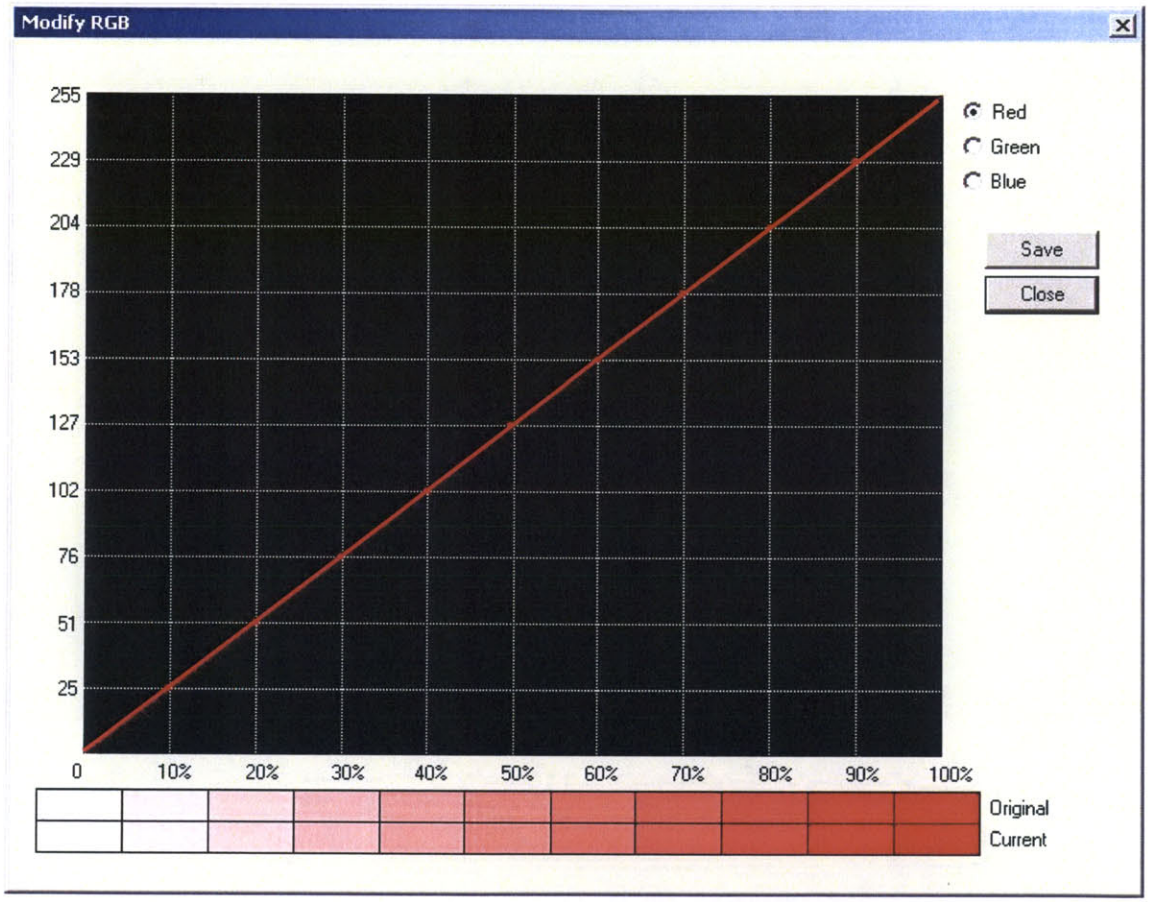

Figure 3.10: Second user-interface design (Design 2)

In this design, the slider bar is replaced by the traditional curve. The user simply selects the color to modify (R, G or B) which shows up in the curve and also at the bottom of the interface. The same feature from Design 1 is incorporated to enable the user to see the difference between the original and current solution. The concern is that the user might think what he sees on the color bar on the screen is equivalent to what appears on paper

which is not really the case. This is, however, a limitation of color control via computer monitors. This may suffice as the user can re-modify the picture quality based on the printed output.

Design 2 was chosen over design 1 as it is more intuitive for color control. Design 1 operates like a digitized system and seems harder to identify with than Design 2. The curve in design 2 makes it obvious that the colors between any two control points are also changed accordingly. Design 2 can be used as a stand-alone application or can be incorporated in the printer property sheets as shown in Design 3 in Figure 3.11.

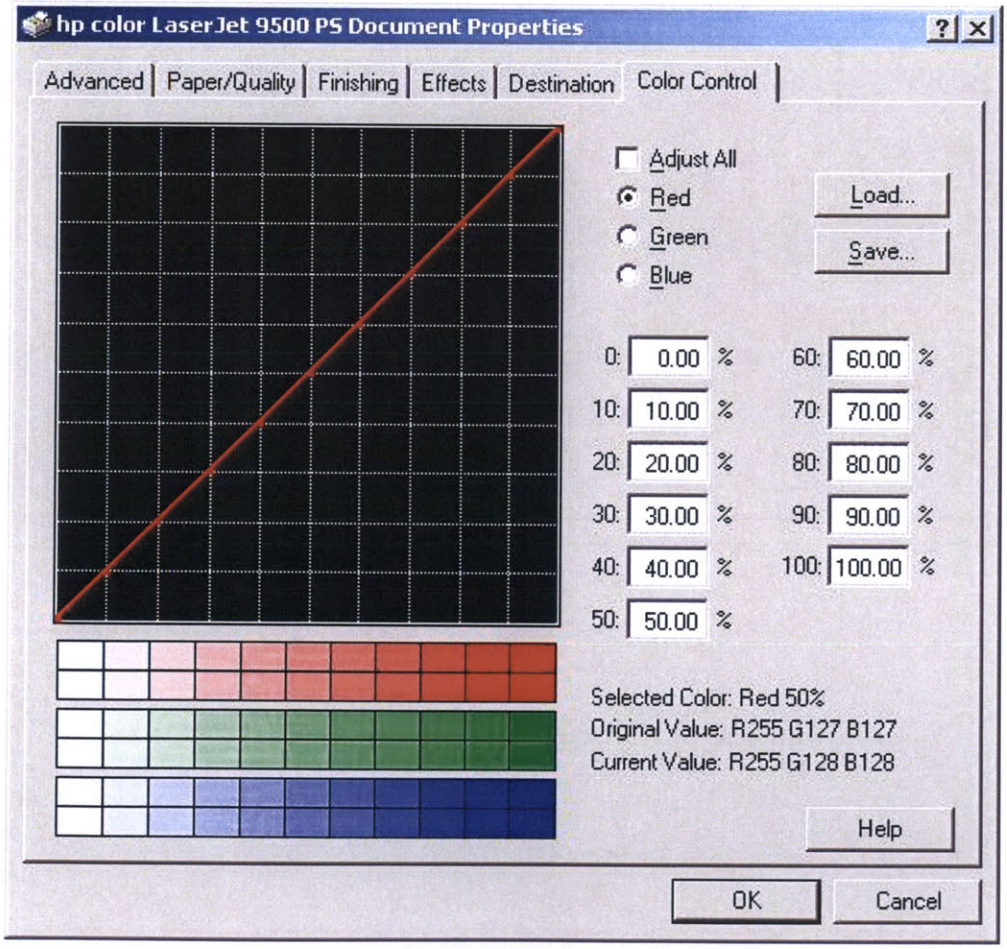

Figure 3.11: Third user-interface design (Design 3)

A few modifications have been made in Design 3 to include all three colors at the base of the curve with the output results shifted to the bottom right corner. Additionally, the user can modify the curve using the input text boxes to the right of the interface instead of dragging the curve to specific points. This might be faster to use as the user becomes more experienced in using the color control curve. The user also has the option of changing all three curves at the same time. Lastly, Help and Load buttons have been included. The Load button should be very useful as users can Load in previous configurations for use or further modification.

3.4 Prototype Workflow

The prototype has been designed for color correction of bitmap images. The prototype is application independent and is linked to a virtual printer. Design 3 is used as the user interface on the property sheet of the virtual printer (BControl(Copy 2)) as shown in Figure 3.12.

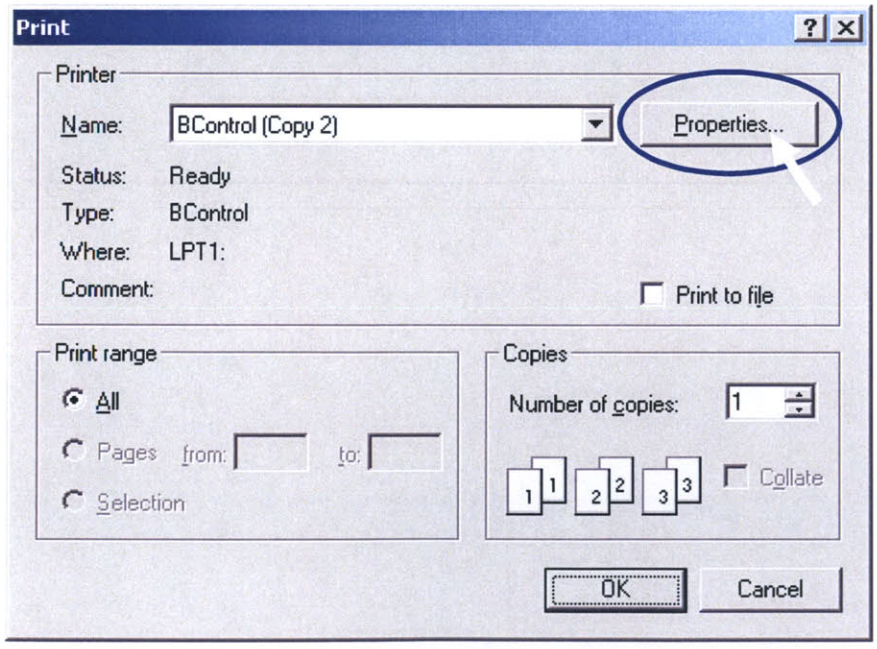

Figure 3.12: Printer dialog and selection of virtual printer

The next step is to click on the 'Properties' button in the printer dialog, which leads to the virtual printer property sheets. The third tab shown in Figure 3.13 has the link to the color control curves which is implemented as a stand-alone DLL application.

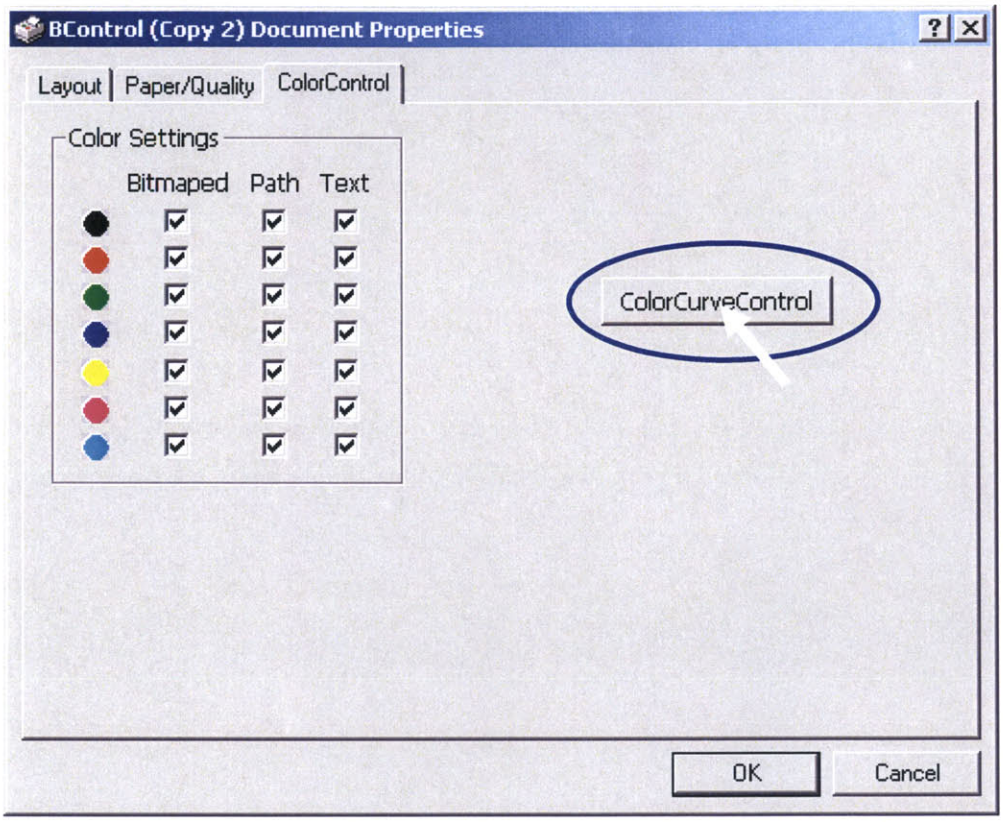

Figure 3.13: Virtual printer property sheet

Clicking on the 'ColorCurveControl' button launches the DLL application shown in Figure 3.14.

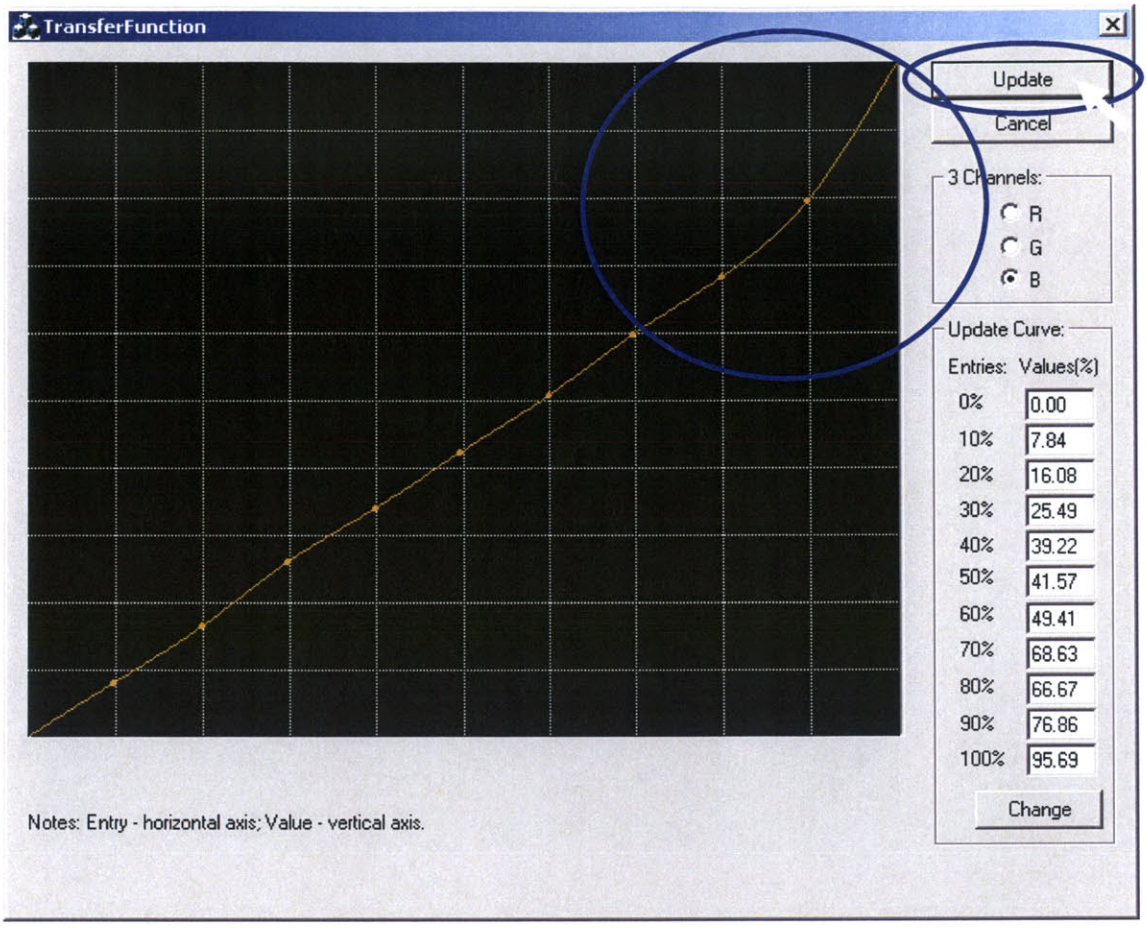

Figure 3.14: DLL stand-alone application for color control

The curve is modified to suit the needs of the bitmap image. In this case, the curve has been modified for the blue color, by lowering the entire curve with greater emphasis in the shadow region (highlighted by the larger circle). Clicking on the update button proceeds to update the files for the color compensation engine. Clicking update loads the dialog shown in Figure 3.15.

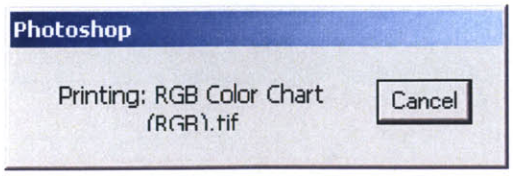

Figure 3.15: Storing of color control values

The following dialog pops up after the values have been stored to select a location for the rendered bitmap image. The user has the option to print the bitmap image directly without first storing it.

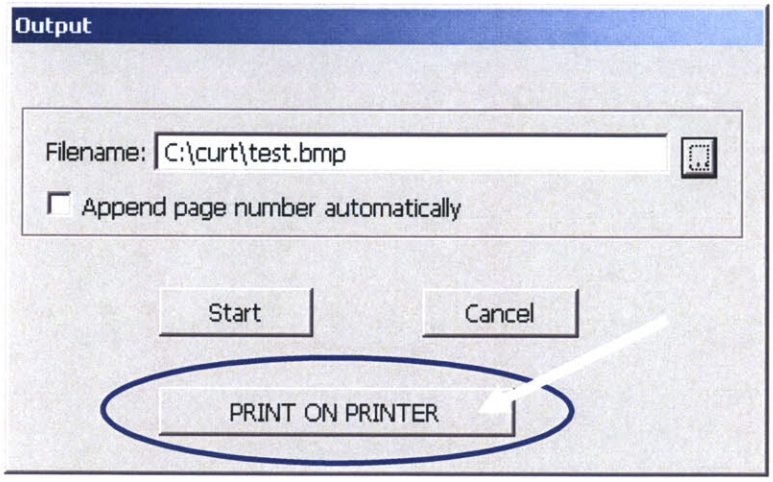

Figure 3.16: Dialog to initiate rendering process

Clicking on the 'Print on Printer' button loads the window shown in Figure 3.17. The user has the option to preview the rendered image before printing as shown.

Figure 3.17: Previewing image before printing

Clicking the 'Print' button after previewing the image loads the regular print dialog shown in Figure 3.18.

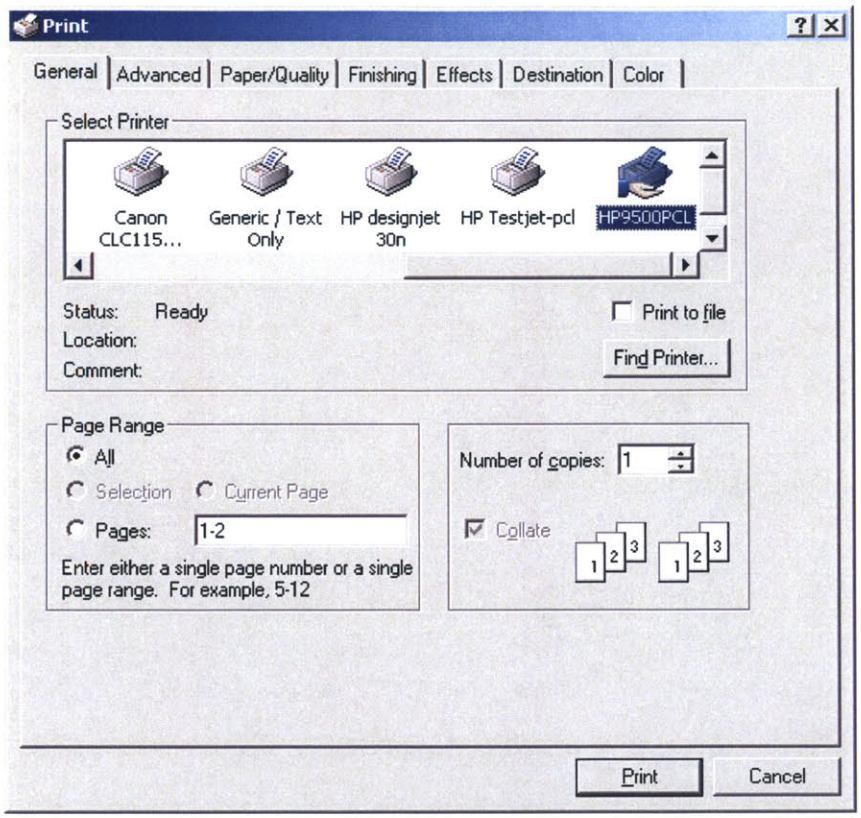

Figure 3.18: Loaded print dialog after image rendering

The user selects the target printer and clicks 'Print'. The prototype workflow is indeed rather long. The module is, however, still in the prototype phase with an aim to develop and prove the workflow. The workflow can be made a lot simpler by by-passing the virtual printer and going straight away to the physical printer as described in the user-interface plug-in section. The simpler approach should certainly be adopted in the final software solution.

3.5 Chapter Summary

The various methods explored in this project have been presented. The merits and demerits of the approaches are analyzed and the choice of the final approach is justified. The details within the developed solution are explained. The user interface designs are presented and the prototype work flow is illustrated. A summary of the entire work flow is given in Figure 3.19.

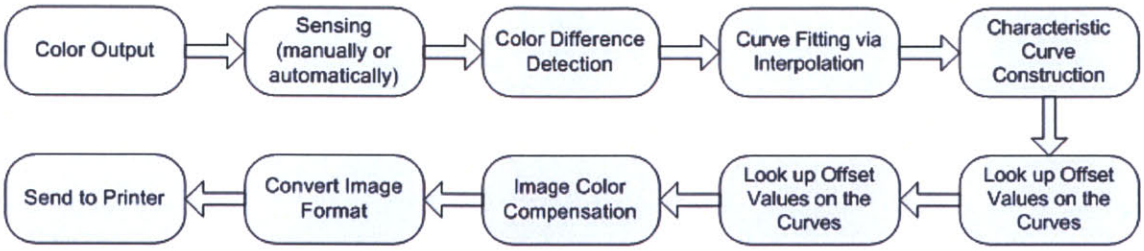

Figure 3.19: Complete workflow of Color optimization solution

Chapter 4 Testing and Evaluation

This section of the thesis demonstrates the tests carried out to validate the effectiveness of the developed prototype. The section explains the procedure involved in carrying out the tests and presents the qualitative and quantitative results of the tests.

To test the effectiveness of the color control module developed, a few tests were designed:

- Qualitative test of picture quality after use of module
- Quantitative test of color differences after use of module
- Qualitative test after change of paper
- Qualitative test after change of printer
- Quantitative test after change of printer

4.1 Test Environment and Procedure

The test requirements are listed as follows:

1) a standard calibration chart
2) a commercial color printer with relatively stable output

The test consists of the following steps:

1) Print a test sheet from Windows system
2) Compare the printout against the standard test sheet: the test sheet is designed to fit behind the standard chart window cutouts. The comparison can be done either visually or with measurement data from a scanner. Usually, the first test sheet would reveal a set of colors differing from the standard one, as shown in Figure 4.1

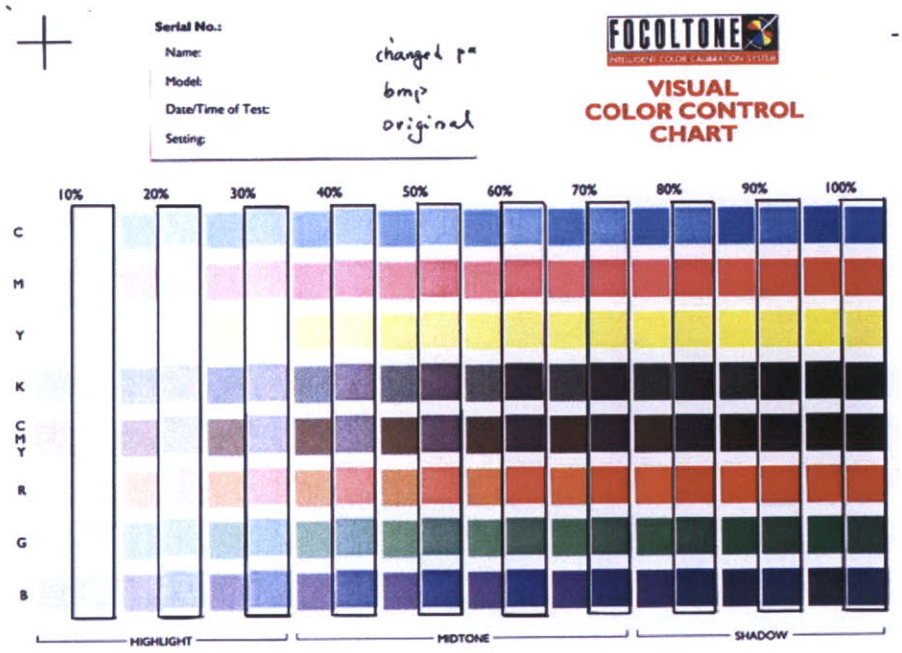

Figure 4.1: Comparing a print out with the calibration chart

Manually, the user simply compares the colors on the standard chart (top colors in the figure) to the colors on the print out (bottom colors in the figure). The user can shift the standard chart either to the left or right dependent on the observed difference with the print out (i.e. the user will be comparing a 20% region on the print out to either a 10% or 30% region on the standard chart). The shifts are in 10% blocks but the user can tell if a 10% shift is too much if the print out turns out to be either too dark or too light in comparison. The user can then decide the magnitude of adjustment required for the curves.

Automatically, a scanner captures the two images (print out and standard chart) as bitmaps. The RGB values for each color block (a few pixels in the middle of the color block) can then be read. The difference can simply be obtained by comparing the two values and the LUT is updated accordingly.

3) Modify primary colors from the UI: once the color differences are detected, the printer characteristic curve is updated by manipulating the curve of each channel from the UI. For instance, if the shadow region of cyan in the print out is too dark in comparison to

the standard chart, the highlight region of the red curve has to be increased as explained in Table 4.2.

4) Compensate color output via the modification of image: based on the characteristic curve gained from the color analysis, the compensation engine automatically calculates the adjustments required and compensates the color data pixel by pixel.

To expound, the compensation engine first receives printer characteristic curves from step 3), and builds a LUT (look-up table) such as the one shown in Table 5.1. This table, converted from the curves, contains 256 items which represent 0-255 color stages. Each row has 4 entities relating color input to the R/G/B output channels. Since the image is stored in the computer as an RGB array, the engine extracts RGB information for each pixel, and looks up the expected input of that pixel for the printer. For example, assuming the image has a pixel with RGB values 3/5/4, according to Table 4.1, the RGB value sent to the printer should be 2/1/2. This process compensates each pixel image by setting a printer input offset and the final rendered image (consisting of the compensated pixels) is saved as a bitmap file.

The image after the compensation usually does not look that accurate on the screen, but once it is printed on the specific printer it was calibrated for, the output turns out great. The detailed implementation of updating the image pixels is covered in Wei's thesis.

Table 4.1: Sample LUT

Input	Output		
Color Index	R Channel	G Channel	B Channel
0	1	2	1
1	2	5	2
2	3	3	4
...	...	...	...
255	255	254	255

5) Reprint the test sheet to verify the improved color results, and re-run the experiment if necessary.

The entire procedure is summarized in Figure 4.2.

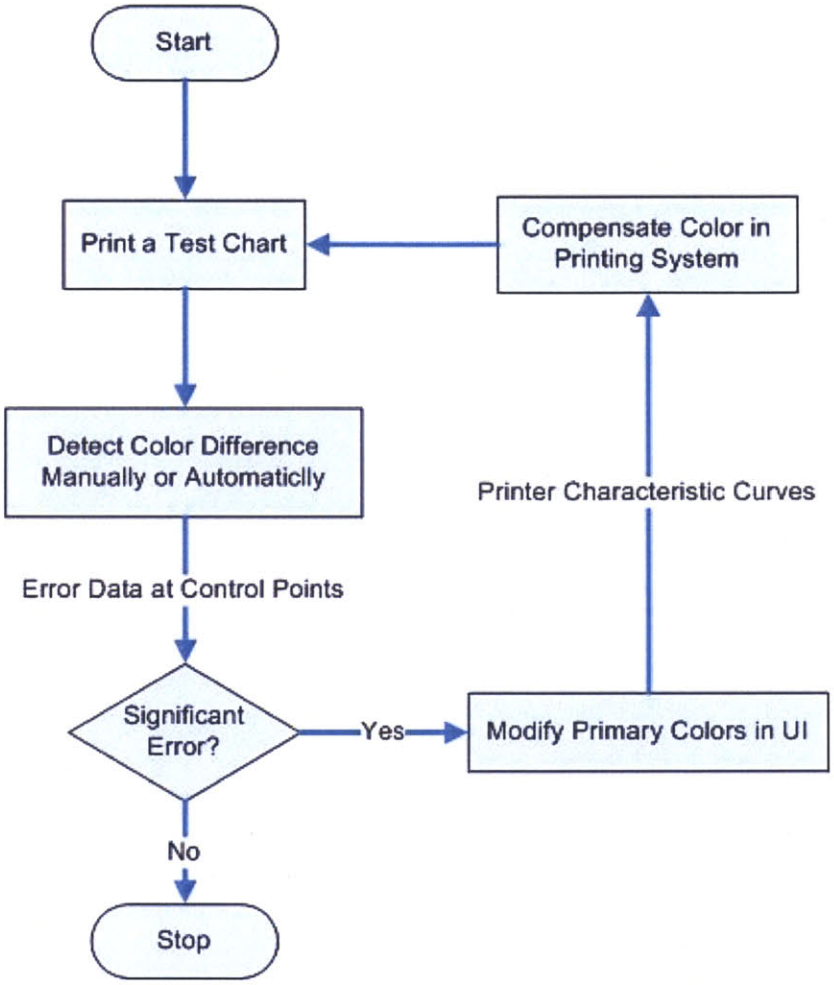

Figure 4.2: Flowchart of test procedure

It should be noted that the output tests that have been copied and pasted in this thesis might look slightly different from the description in the thesis. This is due to the fact that, the printing conditions and paper used for this thesis will be different from those under which the tests were carried out. The pictures in the thesis are electronic copies of the rendered

images that were used for the tests. These are some of the same issues that this project seeks to address.

4.2 Testing via picture quality (Qualitative results)

The main use of the color control module is to modify picture quality via color control. Users will simply look at the output pictures, modify the color control curves and re-print expecting better output quality. As a result, this constituted the first set of major tests that were run for the color control module.

The standard Kikuze test chart in Appendix A was used to carry out the tests. Correcting the picture quality is, however, not a straight forward method. Complications arise due to the fact that the monitor presents its output in RGB format whereas the printer actually prints in CMYK. This module is designed for RGB as users might identify with the RGB format more than with CMYK. In order to correct the color, a transformation therefore has to be made from CMY to RGB to compensate for the printer-monitor color output differences. This is, however, not a full transformation from RGB to CMY, but a simple one for the purposes of printing. This transformation is shown in Table 4.2.

Table 4.2: RGB-CMY transformation

CMY decrease ↓	CMY increase ↑
C (highlight) ↓ = R curve (shadow) ↑	C (highlight) ↑ = R curve (shadow) ↓
C (shadow) ↓ = R curve (highlight) ↑	C (shadow) ↑ = R curve (highlight) ↓
M (highlight) ↓ = G curve (shadow) ↑	M (highlight) ↑ = G curve (shadow) ↓
M (shadow) ↓ = G curve (highlight) ↑	M (shadow) ↑ = G curve (highlight) ↓
Y (highlight) ↓ = B curve (shadow) ↑	Y (highlight) ↑ = B curve (shadow) ↓
Y (shadow) ↓ = B curve (highlight) ↑	Y (shadow) ↑ = B curve (highlight) ↓

The print out of the first picture is shown in Figure 4.3.

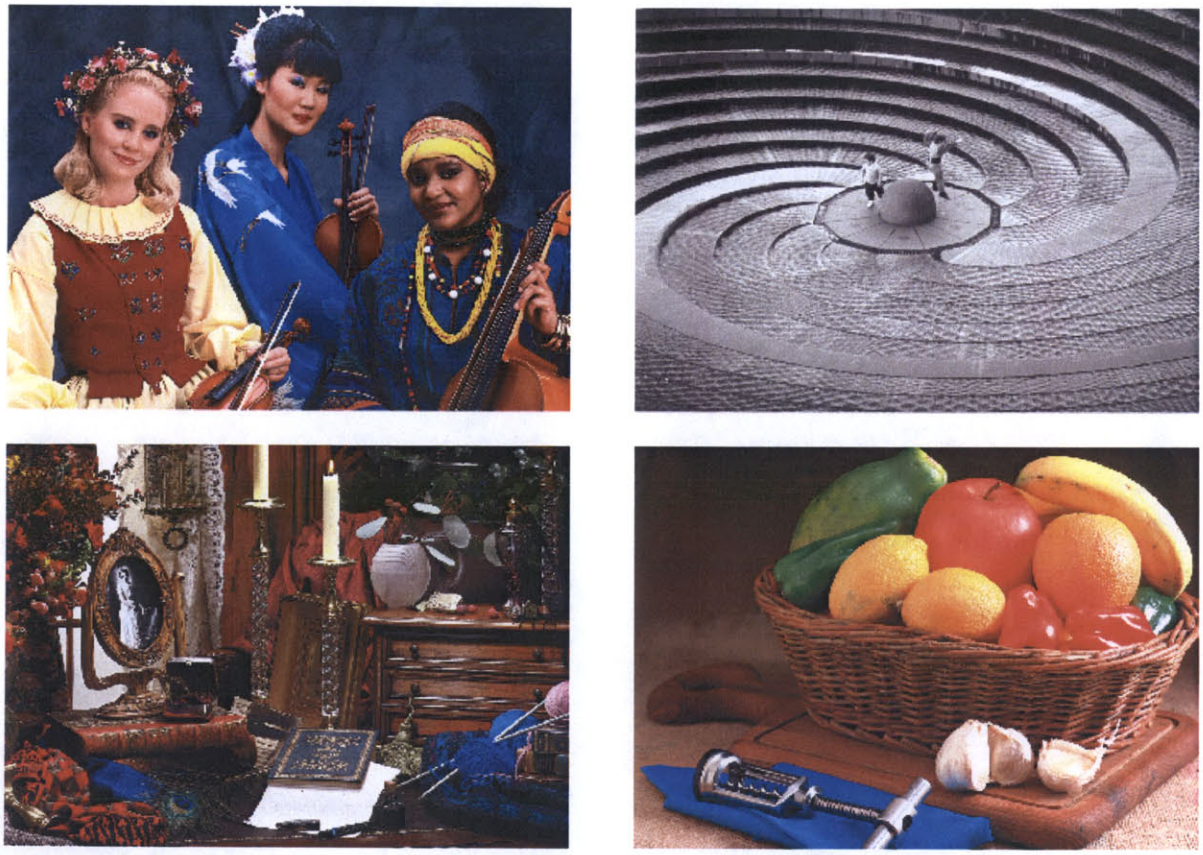

Figure 4.3: Original printed picture with no color compensation [8]

Noticeable on the printout is the fact that the red component, especially in the shadow region, is too high. The fruits in the bottom right picture, for instance, look too colorful to be real. The faces in the top left picture and the dress of the Caucasian lady also appear too reddish. Furthermore, the shadow of the blue color also appears too high. The dark areas in blue are too dark in the first picture.

The red (R) sections of the picture are printed or influenced by the magenta (M) component of the printer. To modify the red color, the user therefore needs to modify the green (G) curve in the color control module. This might seem mind boggling at first but is actually the

observed effect and it works. The red shadow and midtone regions are too high. This implies that the magenta component is too high. The magenta shadow corresponds to the green curve highlight region (shadow in RGB corresponds to highlight in CMYK). A desire to decrease the magenta shadow therefore translates to an increase in the green highlight region due to the inverse relationship. The changes are shown in Figure 4.4.

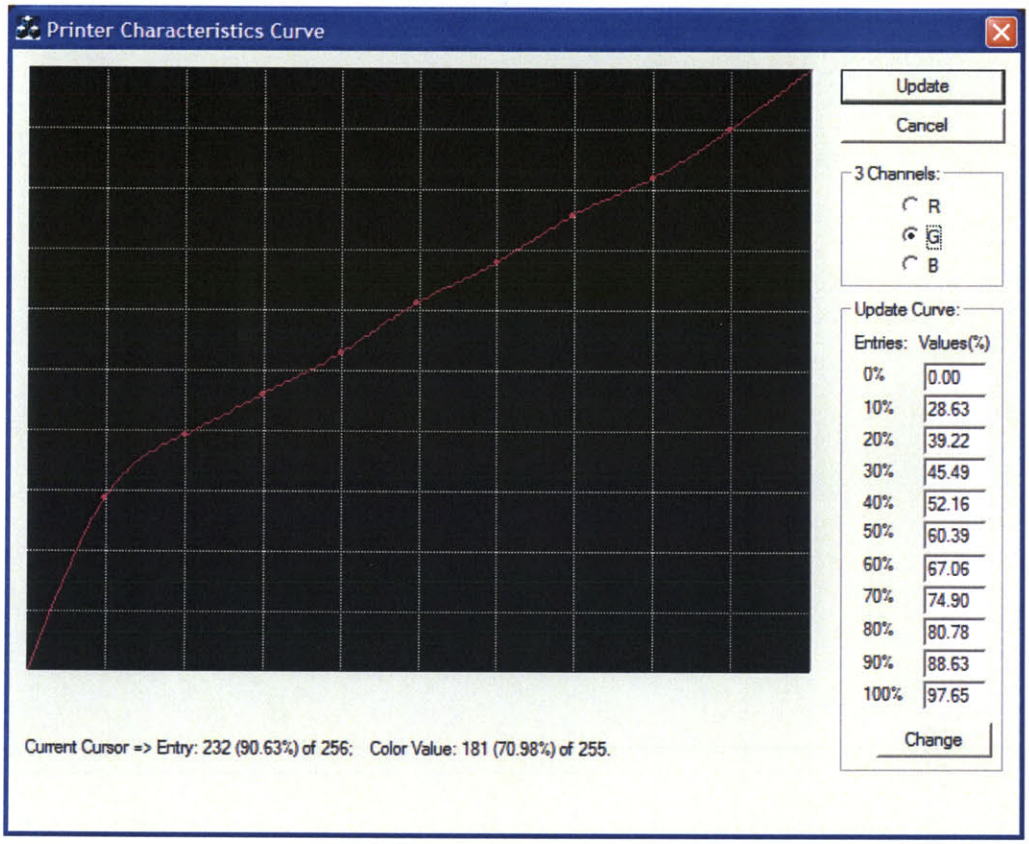

Figure 4.4: Highlight of green curve is modified to reduce shadow of red color in picture

To correct for the dark areas in blue, the red curve is also adjusted accordingly. The changes after correction are shown next to the original image in Figure 4.5.

Original After cycle 1

Figure 4.5: Output picture after first correction

The fruits look more realistic in this picture. However, the entire picture has a whitish tint to it making the ladies slightly pale in complexion. The picture has been over compensated. It is evident from the white tint that the picture needs to be made slightly darker. This implies that all three channels (which make black) need to be modified. The three curves were shifted down to compensate for the white tint and the results are shown in Figure 4.6.

Original After cycle 2

Figure 4.6: Output picture after correction of white tint

This final output gave the desired results. The skin complexion is close to perfect and real. The fruits look a lot more realistic in this picture. The black and white image also looked quite good but is not shown in the figure. The whitish tint has also been significantly reduced. Qualitatively, this printout gives the best result and proves that the workflow and color compensation engine are effective. Indeed this qualitative analysis remains subjective though the team consulted three color experts - Ryan Wong, Dr Du Xian and Dr Zhang to verify the improvements.

4.3 Testing individual color components (Quantitative results)

To verify the calibration effect, a Gretag Macbethi1 Spectrophotometer was used to detect output errors of 50 color patches (the validation set) on the final printout. This instrument gives delta E values, which quantitatively illustrate CIELAB 2000 differences between the printout and the standard chart.

Figure 4.7 shows the average errors at different color density levels in the validation set. The errors in CMY Black are also measured and shown in Figure 4.8 due to its significant effect on the overall color quality. It is noticeable that the midtone errors on both figures are significantly reduced. However, the error reductions at other regions are less effective. This can be explained by the fact that human eyes are not sensitive to color differences at highlight and shadow regions. However, a better result might be obtained if the printer characteristic curves are updated objectively rather than subjectively. For this purpose, a scanner would need to be introduced as a standard measurement device.

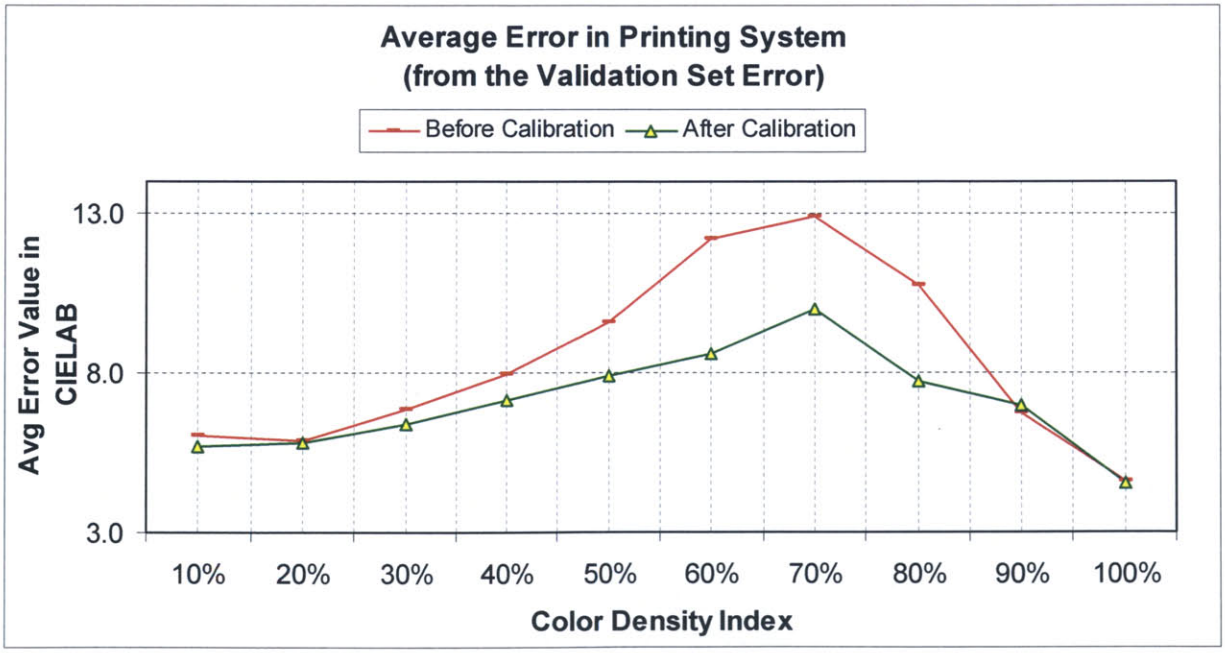

Figure 4.7: The average color error in the printing system
(data from the validation set)

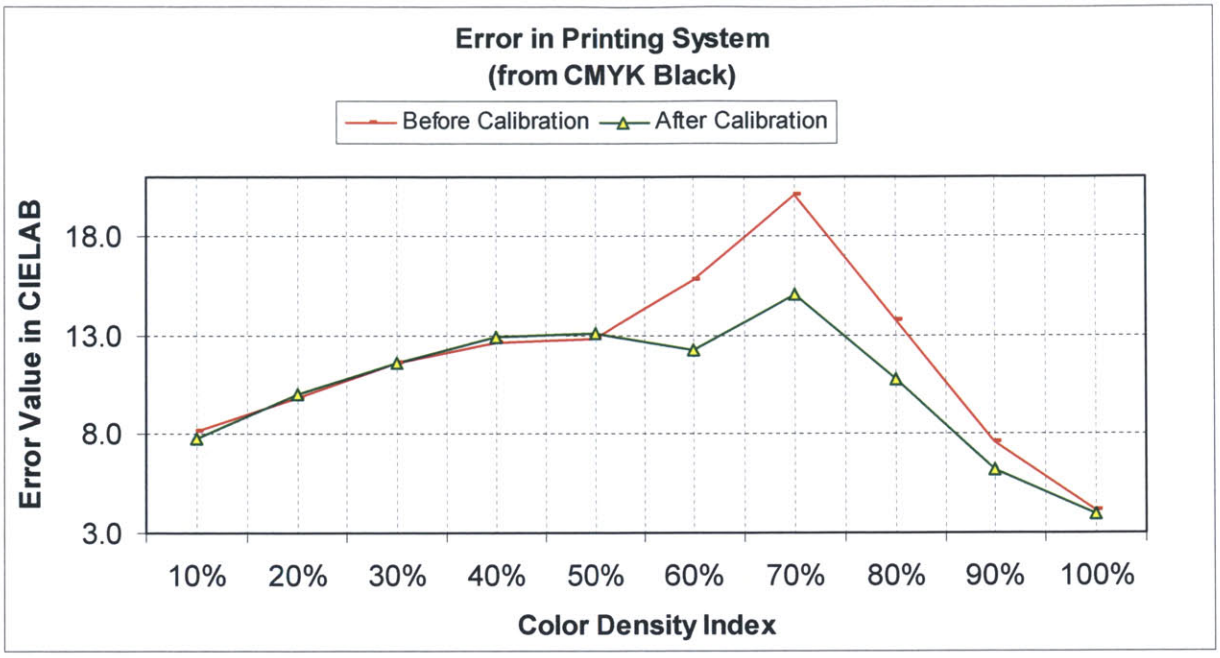

Figure 4.8: The average error of CMY Black in the system

4.4 Qualitative tests after change of paper

To further validate the effectiveness of the Kikuze Color control Module, the paper stock in the HP9500PCL printer was changed. The paper used for the previous tests and for the calibration chart is white in color. In order to test the robustness of our solution, the paper was changed to one which was distinctly different in thickness, surface roughness and color. The paper stock used for these tests was off white in color (hence forth referred to as 'off white paper'). The same procedure described in the previous section was used to carry out the tests.

It should be noted once again that it is rather difficult to fully appreciate the print out quality through this thesis print out. The electronic copies of the rendered image will be meaningless if pasted here as they are electronic and will not exhibit the same qualities seen when they were printed. Instead they will exhibit the effect from printing on the "MIT acid free' paper. The adjustments made to the curves are therefore presented instead of the actual images.

The first print out, which is equivalent to the image shown in Figure 4.3 turned out to be very dark on the off white paper. Interestingly, the colors were too light in the highlight region and too dark in the shadow region. This was understandable given the paper itself appeared darker. The color control curves were therefore adjusted for all three colors by increasing the highlight area and decreasing the shadow area of the curves. This is inversely related to the CMYK output as explained previously. The increase in the highlight region of the RGB curves therefore decreases the effect in the shadow region of the CMYK print out and vice versa for the shadow region of the RGB curves. The changes were made twice and the adjustments are shown in Figure 4.9.

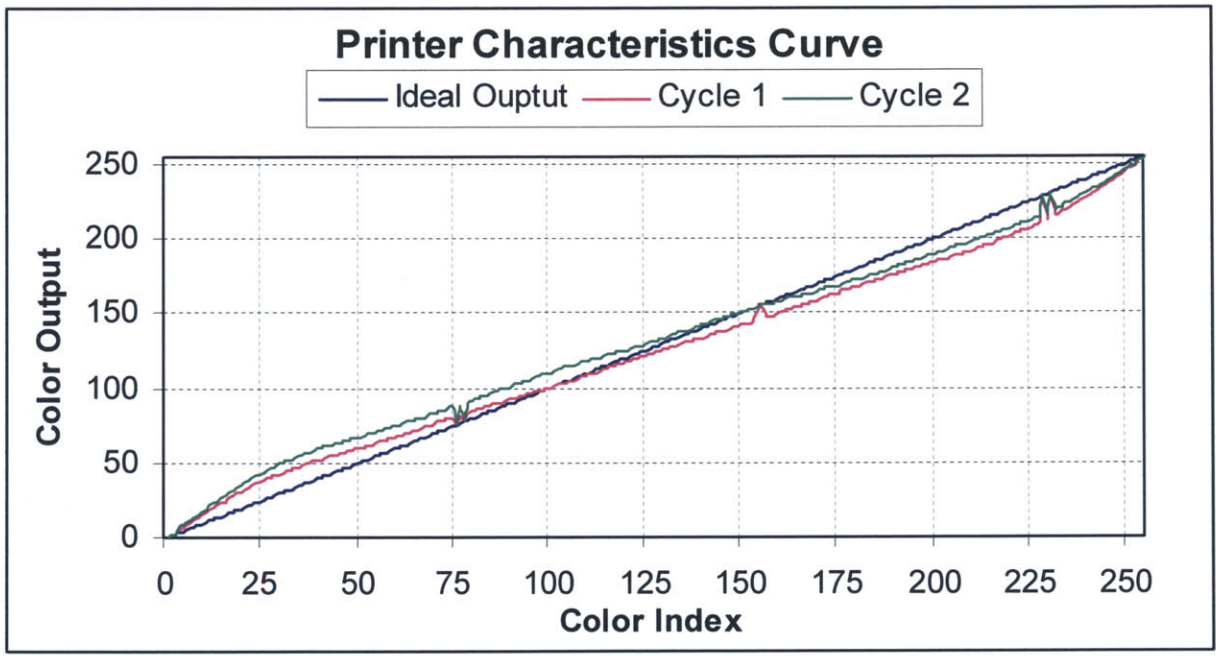

Figure 4.9: Changes in color control curve for off white paper

The change after cycle 1 was not satisfactory. The shadow region had been over compensated and the highlight region had not been compensated enough. The necessary modifications were made, simply by adjusting accordingly evident in the cycle 2 curve in Figure 4.9.

4.5 Qualitative tests after change of printer

To ensure that the module worked with other printers, tests were carried out on a Canon printer with different paper stock. The paper stock was similar to that used in the original tests on the HP printer. The color output was relatively good even for the first print out of the original image in Figure 4.3. However, the faces still exhibited a red tint especially evident in the dress of the Caucasian lady. The shadow and midtone areas of the magenta curve therefore had to be modified. The green curve was adjusted as shown in Figure 4.10.

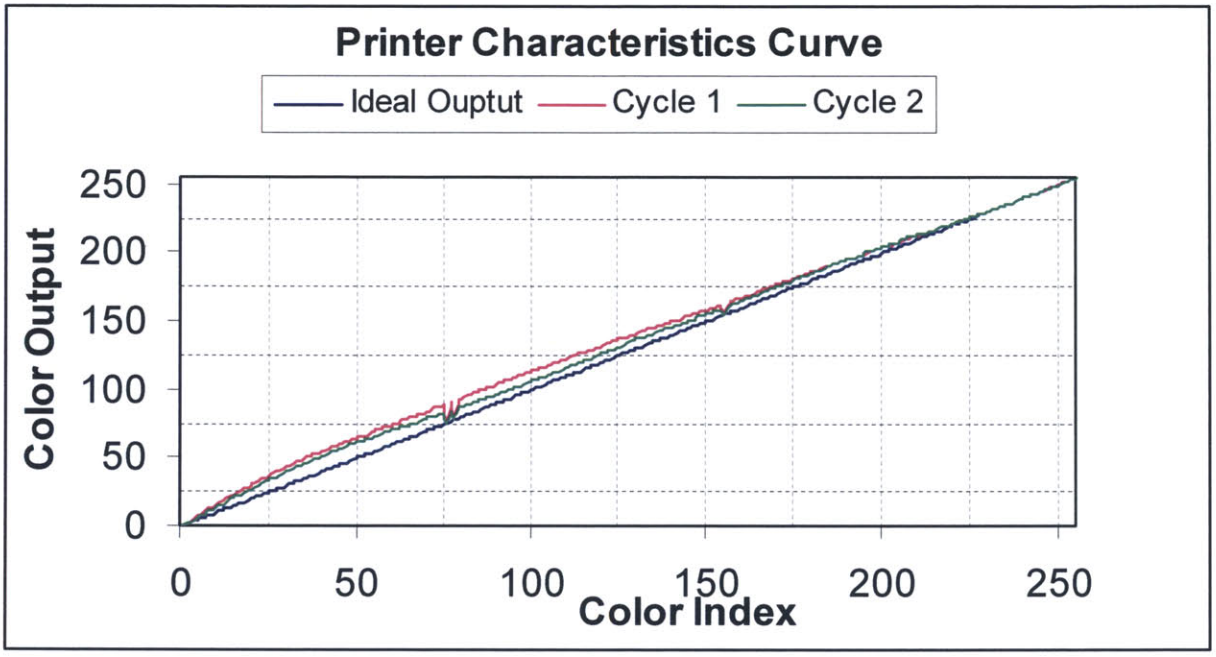

Figure 4.10: Adjustment for red color in Canon print out

Though only two cycles are shown in the figure, the adjustments were done several times due to difficulties in determining how much to change the various curves. The fact that only slight modifications were required also made it difficult and resulted in over compensation in one of the cycles. The effect of over compensation is shown in Figure 4.11.

Original Over compensated

Figure 4.11: Over compensation effects

The green curve was over compensated and created a green tint evident in the pictures.

4.6 Quantitative test after change of printer

The same set of quantitative analysis carried out on the HP9500 printer was carried out on the Canon printer. The CMY results are of interest as they clearly show the impact of the color compensation module. The results are presented in Figure 4.12.

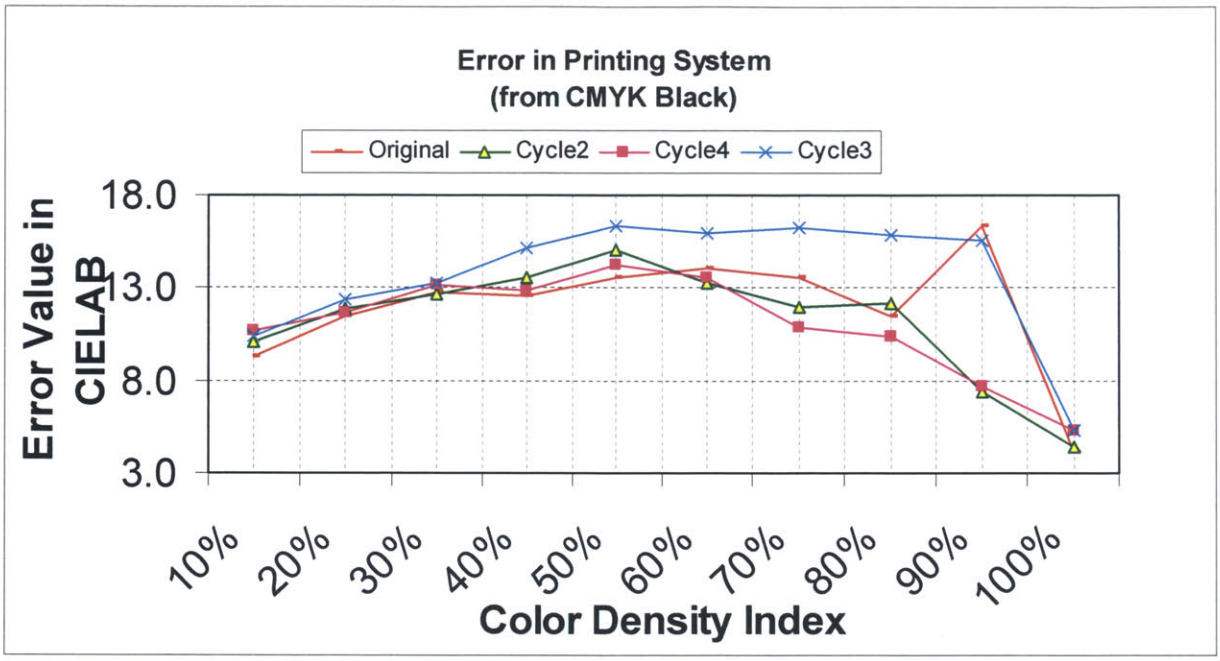

Figure 4.12: Average Errors of CMY Black in the system

The changes in cycles 2 and 4 were only slightly different from the original print out. As stated in the previously, the qualitative output by visual inspection from the Canon printer was already quite good. The graph in Figure 4.12 confirms the visual quality. Cycle 3 differs sharply and is the quantitative representation of the overcompensation shown in Figure 4.11. These are some of the difficulties an inexperienced user might face in carrying out a manual calibration. Cycle 4 produced the best results quantitatively with the main differences evident in the 90% region where the original differed sharply from the calibration chart.

4.7 Chapter Summary

Exhaustive tests of the developed module were carried out and their results presented in this section. The developed module was found to be robust enough for changes in paper stock and changes in printer. Both the qualitative and quantitative results proved the effectiveness of the module.

Chapter 5 Conclusions and Recommendations

The key project objectives at the start of the development of the color management system for RGB printing are shown below. The tasks that were achieved during the project have been checked.

- ✓ Figure out the optimal and feasible stage for integration of the software solution with the Windows XP printing architecture (key entry point)
- ✓ Use RGB color space instead of CMYK in the existing solution
- ✓ Determine work flow of color control module
- ✓ Develop and integrate color control module with the existing Kikuze solution
- ✓ Test and evaluate work flow and developed solution
- ✓ Design user interface
- ✓ Develop full working prototype

  Make prototype ready for demos to potential customers

  Extend prototype for automatic color correction

Evident from the check-list above is a fully developed, tested and working prototype with complete and successful color management. The prototype is almost ready for demonstration to potential customers though that the objective has not been checked. The module needs further development for automatic color correction. The basic steps would involve reading the scanned data, noting the differences between the standard chart and the printed output and correcting the color control curves accordingly. The programming task involved should be relatively simple, however, the challenge will be determining which colors and at which points the adjustments should be made.

Based on the test results and analysis, the project can be judged as a success. A few issues still need to be addressed and are expressed as follows:

RGB versus CMYK

The correction of color in the current module can be a little confusing. The color transition from RGB to CMYK from monitor to printer introduces complications in figuring out which curves to modify. In the current module for instance, to modify the red color, the green curve needs to be altered. This is counter intuitive and will leave many users confused. To make the color control module more user-friendly, we suggest that the color control module shift from displaying RGB curves to CMY curves to match the printer output. The color compensation engine can still be implemented with RGB data though the user will be presented with CMY curves. Though this would not represent a one-to-one mapping from color control module to printer ink dispensation, it would certainly be closer than the green curve mapped to magenta to modify the red color in the print outs. This would be a lot easier for users to identify with as they can use their intuition to determine which color curves need modification.

Bitmap size

The size of the rendered image is usually large as it is output in bitmap format. Though this does not affect the rendering processing time, it affects the printing resources and printing time. This becomes a nuisance if lots of pictures need to be printed for a book or report for instance. Compression techniques should therefore be employed in the module to reduce the bitmap size and speed up the printing process of rendered images. Such technology has been used by Zenographics, and Kikuze can collaborate with Zenographics to develop this solution.

Printing other images besides bitmaps

The current solution needs to be further developed to handle other image types besides bitmaps. Many pictures are printed from other applications such as Word, PowerPoint and Excel, in addition to the numerous PDF files printed. Since the module is application independent, it should be further developed to either render picture images embedded in the different file types or render the entire file.

Automatic color correction

The development of the automatic module should be thoroughly tested. In the current development, it was noticed that the qualitative analysis showed greater improvement (though it was subjective) compared to the measured data. This problem would probably surface during automatic calibration as well, to the extent that the automatically corrected picture might not look as good to the human eye as expected.

References

[1] Du, X. "Optimization of Color Print Process Control" PhD Thesis in IMST, SMA, 2005

[2] Windows Driver Development Kit Documentation

[3] Dong, W. "Innovative Color management method for RGB printing" MS Thesis in Mechanical Engineering, SMA, 2006

[4] Windows Universal print driver, http://www.paulyao.com/resources/whitepapers/WinHecWP2.html (accessed on 3/17/2006)

[5] Fraser, B., Murphy, C., Bunting, F., "Real World Color Management", Peachpit Press, 2004, page 19-21

[6] Mohideen, S. S., Rajagopalan, S., Wu, Q. "Color Management and Control based on Image Color Data", MS Thesis in Mechanical Engineering, SMA, 2005

[7] Color transformation and the Color matrix, http://www.c-sharpcorner.com/Code/2004/April/Transformations05.asp (accessed on ...)

[8] Focoltone Visual Color Control Chart, Test and Designs by Kikuze Solutions

[9] Microsoft MSDN Library 2000

APPENDIX A Test Chart